Date=12/11/2020
Lecture By=Manish Mahant
Subject ⇒React

IN PREVIOUS LECTURE **(QUICK RECAP) Date-11/11/2020**	**In Today's Lecture (Overview)**
What Is Functional Programming?? Side Effects Conclusion Questions For Self-Practice	What is React??? Hello React Use cases How does React use JavaScript? Setting up your first React app Questions For Self-Practice // CC For The Day

What is React???

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

Hello React

As its official tagline states, React is a library for building user interfaces. React is not a framework – it's not even exclusive to the web. It's used with other libraries to render to certain environments. For instance, React Native can be used to build mobile applications; React 360 can be used to build virtual reality applications; and there are other possibilities besides.

To build for the web, developers use React in tandem with ReactDOM. React and ReactDOM are often discussed in the same spaces as — and utilized to solve the same problems as — other true web development frameworks. When we refer to React as a "framework", we're working with that colloquial understanding.

React's primary goal is to minimize the bugs that occur when developers are building UIs. It does this through the use of components — self-contained, logical pieces of code that describe a portion of the user interface. These components can be composed together to create a full UI, and React abstracts away much of the rendering work, leaving you to concentrate on the UI design.

Use cases

Unlike the other frameworks covered in this module, React does not enforce strict rules around code conventions or file organization. This allows teams to set conventions that work best for them, and to adopt React in any way they would like to. React can handle a single button, a few pieces of an interface, or an app's entire user interface.

While React *can* be used for small pieces of an interface, it's not as easy to "drop into" an application as a library like jQuery, or even a framework like Vue — it is more approachable when you build your entire app with React.

In addition, many of the developer-experience benefits of a React app, such as writing interfaces with JSX, require a compilation process. Adding a compiler like Babel to a website makes the code on it run slowly, so developers often set up such tooling with a build step. React arguably has a heavy tooling requirement, but it can be learned.

This article is going to focus on the use case of using React to render the entire user interface of an application, using tooling provided by Facebook's own create-react-app tool.

How does React use JavaScript?

React utilizes features of modern JavaScript for many of its patterns. Its biggest departure from JavaScript comes with the use of JSX syntax. JSX extends JavaScript's syntax so that HTML-like code can live alongside it. For example:

```
const heading = <h1>Mozilla Developer Network</h1>;
```

This heading constant is known as a **JSX expression**. React can use it to render that `<h1>` tag in our app.

Suppose we wanted to wrap our heading in a `<header>` tag, for semantic reasons? The JSX approach allows us to nest our elements within each other, just like we do with HTML:

```
const header = (
  <header>
    <h1>Mozilla Developer Network</h1>
  </header>
);
```

Of course, your browser can't read JSX without help. When compiled (using a tool like Babel or Parcel), our header expression would look like this:

```
const header = React.createElement("header", null,
  React.createElement("h1", null, "Mozilla Developer Network")
);
```

Setting up your first React app

There are many ways to use React, but we're going to use the command-line interface (CLI) tool create-react-app, as mentioned earlier, which expedites the process of developing a React application by installing some packages and creating some files for you, handling the tooling described above.

It's possible to add React to a website without create-react-app by copying some `<script>` elements into an HTML file, but the create-react-app CLI is a common starting point for React applications. Using it will allow you spend more time building your app, and less time fussing with setup.

create-react-app takes one argument: the name you'd like to give your app. create-react-app uses this name to make a new directory, then creates the necessary

files inside it. Make sure you `cd` to the place you'd like your app to live on your hard drive, then run the following in your terminal:

Initializing your app

create-react-app takes one argument: the name you'd like to give your app. create-react-app uses this name to make a new directory, then creates the necessary files inside it. Make sure you `cd` to the place you'd like your app to live on your hard drive, then run the following in your terminal:

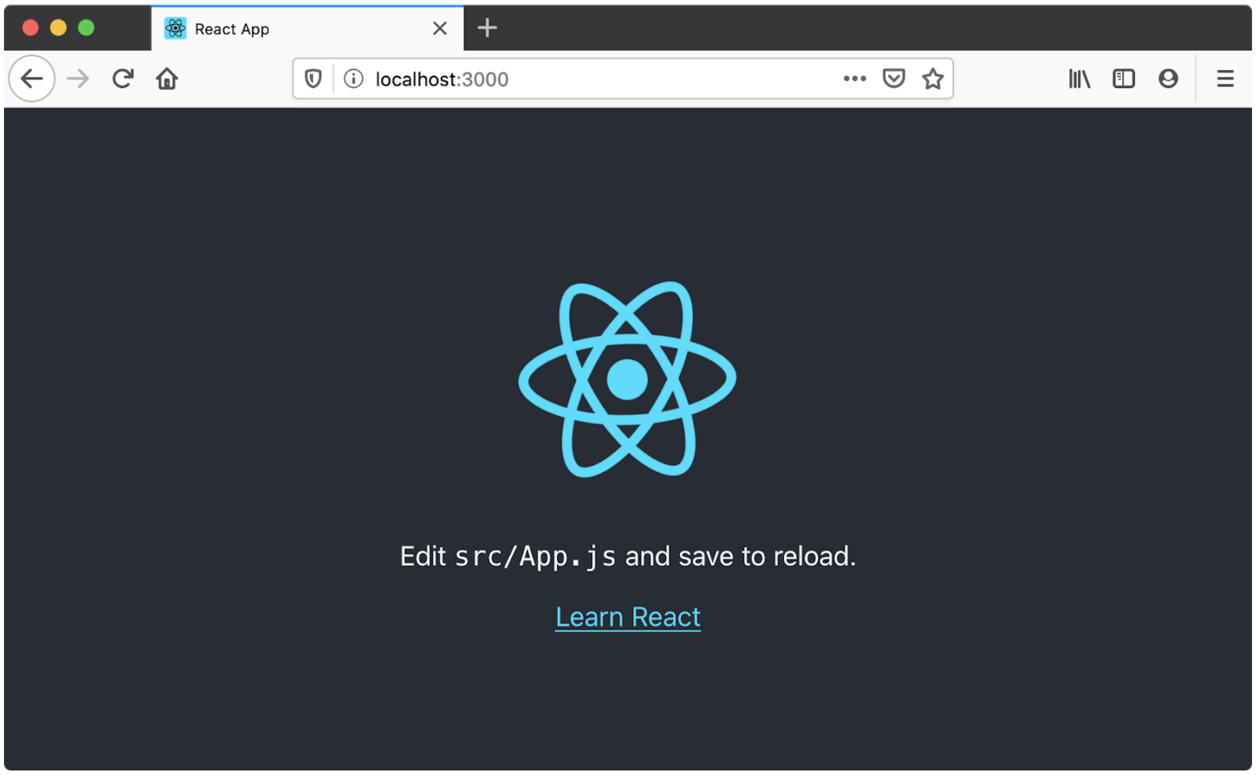
```
npx create-react-app moz-todo-react
```

This creates a `moz-todo-react` directory, and does several things inside it:

- Installs some npm packages essential to the functionality of the app.
- Writes scripts for starting and serving the application.
- Creates a structure of files and directories that define the basic app architecture.
- Initializes the directory as a git repository, if you have git installed on your computer.

create-react-app will display a number of messages in your terminal while it works; this is normal! This might take a few minutes, so now might be a good time to go make a cup of tea.

When the process is complete, `cd` into the `moz-todo-react` directory and run the command `npm start`. The scripts installed by create-react-app will start being served at a local server at localhost:3000, and open the app in a new browser tab. Your browser will display something like this:



Edit `src/App.js` and save to reload.

Learn React

Application structure

```
moz-todo-react ├── README.md ├── node_modules ├── package.json
├── package-lock.json ├── .gitignore ├── public │ ├──
favicon.ico │ ├── index.html │ └── manifest.json └── src ├──
App.css ├── App.js ├── App.test.js ├── index.css ├── index.js
├── logo.svg └── serviceWorker.js
```

The **`src`** directory is where we'll spend most of our time, as it's where the source code for our application lives.

The **`public`** directory contains files that will be read by your browser while you're developing the app; the most important of these is `index.html`. React injects your code into this file so that your browser can run it. There's some other markup that helps create-react-app function, so take care not to edit it unless you know what you're doing. You very much should change the text inside the `<title>` element in this file to reflect the title of your application. Accurate page titles are important for accessibility!

The `public` directory will also be published when you build and deploy a production version of your app. We won't cover deployment in this tutorial, but you should be able to use a similar solution to that described in our Deploying our app tutorial.

To know More About React Refer to the Video Below
https://www.youtube.com/watch?v=Ke90Tje7VS0
Website
https://reactjs.org/

Questions For Self-Practice // CC For The Day

https://au-assignment.s3.ap-south-1.amazonaws.com/Week_20_Day_4_Challenge-f8d35923-be19-4cc6-8f4c-300341f05e22.pdf